On the Deterministic Complexity of Factoring Polynomials over Finite Fields

Victor Shoup
Computer Sciences Department
University of Wisconsin–Madison
Madison, WI 53706

February 21, 1989

Abstract. We present a new deterministic algorithm for factoring polynomials over $\mathbf{Z}_p$ of degree n. We show that the worst-case running time of our algorithm is $O(p^{1/2}(\log p)^2 n^{2+\epsilon})$, which is faster than the running times of previous deterministic algorithms with respect to both n and p. We also show that our algorithm runs in polynomial time for all but at most an exponentially small fraction of the polynomials of degree n over $\mathbf{Z}_p$. Specifically, we prove that the fraction of polynomials of degree n over $\mathbf{Z}_p$ for which our algorithm fails to halt in time $O((\log p)^2 n^{2+\epsilon})$ is $O((n \log p)^2/p)$. Consequently, the average-case running time of our algorithm is polynomial in n and $\log p$.

Keywords: factorization, finite fields, irreducible polynomials.

This research was supported by NSF grants DCR-8504485 and DCR-8552596.

Appeared in *Information Processing Letters* 33, pp. 261–267, 1990.

An preliminary version of this paper appeared as University of Wisconsin–Madison, Computer Sciences Department Technical Report #782, July 1988.

1. Introduction

Consider the problem of factoring a polynomial of degree n in $\mathbf{Z}_p[X]$ where p is prime. There are several *deterministic* algorithms for this problem whose running time is polynomial for small p, i.e. polynomial in n and p. One of the asymptotically fastest deterministic algorithms is that of Berlekamp [5] as refined by von zur Gathen [12]. The Berlekamp-von zur Gathen algorithm uses $O(M(n) + pn^{2+\epsilon})$ arithmetic operations in $\mathbf{Z}_p$, where $M(n)$ is the number of operations required to multiply two n by n matrices. Currently, the best known upper-bound on $M(n)$ is approximately $O(n^{2.4})$ [11]. In this paper, the expression n^ϵ denotes a fixed, but unspecified, polynomial in $\log n$.

There are also many *probabilistic* algorithms for this problem whose *expected* running time is polynomial, i.e. polynomial in n and $\log p$. One of the asymptotically fastest probabilistic algorithms is due to Ben-Or [4]. Ben-Or's algorithm uses $O((\log p)n^{2+\epsilon})$ expected operations in $\mathbf{Z}_p$. The running time of a different probabilistic algorithm due to Cantor and Zassenhaus [9] is also $O((\log p)n^{2+\epsilon})$.

This state of affairs suggests that there may be a significant gap between the deterministic and probabilistic complexity of this problem. Indeed, the running time of Ben-Or's probabilistic algorithm improves upon the running time of the Berlekamp-von zur Gathen deterministic algorithm with respect to both p and n. With respect to p, this improvement is exponential ($\log p$ vs. p), and it gives us an algorithm that runs in polynomial time for large p. With respect to n, this improvement is only polynomial ($n^{2+\epsilon}$ vs. $n^{2.4}$), but it could be substantial in cases where p is very small (e.g. $p = 2$) and n is very large.

In this paper, we show that this gap is not quite so large. We present a new deterministic algorithm for factoring polynomials of degree n in $\mathbf{Z}_p[X]$, and prove the following results about its running time (expressed in terms of operations in $\mathbf{Z}_p$):

(1) The worst-case running time is $O(p^{1/2}(\log p)^2 n^{2+\epsilon})$.

(2) The fraction of polynomials of degree n over $\mathbf{Z}_p$ for which our algorithm fails to halt in time $O((\log p)^2 n^{2+\epsilon})$ is $O((n \log p)^2/p)$.

Result (1) is significant for a couple of reasons. First, if p is very small, the dependence on n in the running time becomes the dominant factor. The Berlekamp-von zur Gathen algorithm requires the triangularization of an n by n matrix, and hence the $M(n)$ term in the running time. Ben-Or's algorithm avoids the need to do this linear algebra by using randomization. Our algorithm also avoids the need to do any linear algebra, but without resorting to randomization.

Second, if p is very large, the dependence on p in the running time becomes the dominant factor. The Berlekamp-von zur Gathen algorithm requires a deterministic search through potentially all of $\mathbf{Z}_p$. Ben-Or's algorithm replaces this brute-force search with a fast random search. Our algorithm performs a deterministic search, but we prove that the length of this search is bounded by $p^{1/2}(\log p)$. The dependence on p in our algorithm, though exponentially worse than that of Ben-Or's, is still significantly better than that of the Berlekamp-von zur Gathen algorithm.

Result (2) is of interest for a couple of reasons. First, it shows that our algorithm runs in polynomial time for all but at most an exponentially small fraction of polynomials of degree n over $\mathbf{Z}_p$. One could conjecture that our algorithm in fact runs in polynomial time for all inputs, but proving this appears to be very hard; our result at least gives some quantitative evidence in support of such a conjecture.

Second, this result, in conjunction with result (1), implies that the average-case running time of our algorithm is polynomial in n and $\log p$, assuming the input is chosen from a uniform distribution on all polynomials over $\mathbf{Z}_p$ of degree n. There are very few problems in computational number theory whose average-case complexity have been analyzed. For example, Knuth and Trabb Pardo [14] and Hafner and McCurley [13] have analyzed the average-case complexity of factoring over the integers; Collins [10] has analyzed the average-case complexity of an algorithm for factoring polynomials over the rationals. We are unaware of any previous work analyzing the average-case complexity of factoring polynomials over finite fields.

Our algorithm is fairly simple, and the space requirement of our algorithm is polynomial. The analysis of the dependence on n in the running time of our algorithm relies on fast algorithms for multiplying polynomials over a ring. The worst-case and average-case analysis of the dependence on p in the running time of our algorithm makes use of estimates of the number of solutions to equations over finite fields; similar techniques have been previously used in the analysis of various probabilistic algorithms [2, 3, 4].

The rest of this paper is organized as follows: Section 2 describes our new factoring algorithm, Section 3 analyzes its worst-case complexity, and Section 4 analyzes its average-case complexity. One last matter of notation: throughout this paper, $\log x$ denotes the logarithm of x to the base 2.

2. A New Factoring Method

In this section, we describe a new algorithm for factoring polynomials over $\mathbf{Z}_p$.

Let $f \in \mathbf{Z}_p[X]$ be a polynomial of degree n that we wish to factor. As in Ben-Or [4] and Cantor and Zassenhaus [9], we first perform distinct degree factorization. That is, we construct polynomials $f^{(1)}, \ldots, f^{(n)}$ where $f^{(d)}$ $(1 \leq d \leq n)$ is the product of all the distinct monic irreducible polynomials of degree d that divide f.

Let $1 \leq d \leq n$ be fixed and let $g = f^{(d)}$. We want to factor g. Suppose $g = g_1 \cdots g_k$, where the g_i's are distinct monic irreducible polynomials of degree d. We can assume that $k > 1$. Let $m = \deg g = kd$. Also, let $R = \mathbf{Z}_p[X]/(g)$ and $x = X \bmod g \in R$. Finally, let θ_i be the natural homomorphism from the ring R onto the field $\mathbf{Z}_p[X]/(g_i)$.

The well-known *Berlekamp subalgebra* B of R is defined by $B = \{\alpha \in R : \alpha^p = \alpha\}$. Equivalently, we have $B = \{\alpha \in R : \theta_i(\alpha) \in \mathbf{Z}_p \text{ for each } i = 1, \ldots, k\}$. Following Camion [7], we call a subset $S \subset B$ a *separating set* if for any $1 \leq i < j \leq k$ there exists $s \in S$ such that $\theta_i(s) \neq \theta_j(s)$.

Many factoring algorithms, including those of Berlekamp [5] and Camion [7], involve the computation of a separating set. Berlekamp's algorithm uses as a separating set a $\mathbf{Z}_p$-basis for the Berlekamp subalgebra B.

Camion's algorithm uses as a separating set $\{T(x), T(x^2), \ldots, T(x^{2d-1})\}$, where T is the so-called "McEliece operator" defined by $T(\alpha) = \alpha + \alpha^p + \cdots + \alpha^{p^{d-1}}$. In terms of the dependence on n, the computation of a separating set is the bottleneck in these algorithms.

Our algorithm constructs a separating set in the following way. We compute the coefficients of $h(Y) \in R[Y]$ where $h(Y) = (Y - x)(Y - x^p) \cdots (Y - x^{p^{d-1}})$. Suppose $h(Y) = h_0 + \cdots + h_{d-1}Y^{d-1} + Y^d$. Then $\{h_0, \ldots, h_{d-1}\}$ is a separating set. This follows from the fact that $h^{\theta_i}(Y) = (Y - (x^{\theta_i}))(Y - (x^{\theta_i})^p) \cdots (Y - (x^{\theta_i})^{p^{d-1}}) = g_i(Y)$ for $i = 1, \ldots, k$ and the g_i's are distinct.

Now that we have a separating set $S = \{h_0, \ldots, h_{d-1}\}$, we can proceed to factor g as follows. We construct finer and finer partial factorizations $U \subset \mathbf{Z}_p[X]$ consisting of monic polynomials with $\prod_{u \in U} u = g$. Initially, we put $U = \{g\}$. We make use of the operation $Refine(U, v)$, which, when given a partial factorization U and a polynomial $v \in \mathbf{Z}_p[X]$, produces the refinement of U given by $\bigcup_{u \in U}\{\gcd(u, v), u/\gcd(u, v)\} - \{1\}$. To obtain a complete factorization of g, we proceed as follows. For $z = 0, 1, \ldots,$ until $|U| = k$, we execute the following *refinement step:*

> For each s in the separating set S, replace U with $Refine(U, s + z)$, and then, if p is odd, replace U with $Refine(U, (s + z)^{(p-1)/2} - 1)$.

We omit a rigorous proof of the correctness of our algorithm, which follows easily from the fact the S is a separating set.

3. Worst-Case Analysis

In this section, we analyze the worst-case complexity of our algorithm. We shall prove

Theorem 1. *Let f be a polynomial of degree n in $\mathbf{Z}_p[X]$. Then our algorithm will completely factor f using $O(p^{1/2}(\log p)^2 n^{2+\epsilon})$ operations in $\mathbf{Z}_p$.*

We will make use of the following results concerning the complexity of performing polynomial arithmetic.

Lemma 3.1. *Let R be a commutative ring with unity, and let F be a field. Let $L(n) = \log n \log\log n$.*

(1) Multiplication of two polynomials in $R[X]$ of degree $\leq n$ can be performed using $O(nL(n))$ operations ($+, -, \times$ only) in R.

(2) Let $\alpha_1, \ldots, \alpha_n \in R$. Then the coefficients of $(X - \alpha_1) \cdots (X - \alpha_n) \in R[X]$ can be computed using $O(nL(n)(\log n))$ operations ($+, -, \times$ only) in R.

(3) Let f and g be polynomials in $F[X]$ of degree $\leq n$, $g \neq 0$. Then $f \bmod g$ can be computed using $O(nL(n))$ operations in F.

(4) Let $f, g_1, \ldots, g_k$ be polynomials in $F[X]$ such that $\deg f \leq n$ and $\deg g_1 + \cdots + \deg g_k \leq n$. Then $f \bmod g_1, \ldots, f \bmod g_k$ can be computed using $O(nL(n)(\log k))$ operations in F.

(5) Let f and g be polynomials in $F[X]$ of degree $\leq n$. Then the greatest common divisor of f and g can be computed using $O(nL(n)(\log n))$ operations in F.

(1) is proved in Cantor and Kaltofen [8]. We note that the results of Schönhage [17] would actually be sufficient for our purposes. (2) follows from (1) by a divide and conquer method (see Borodin and Munro [6, p. 100]). (3) follows from (1) by a Newton iteration scheme (see Borodin and Munro [6, p. 95]). (4) follows from (3) by a divide and conquer method (see Borodin and Munro [6, p. 100]). (5) follows from (1) by an algorithm described in Aho, Hopcroft and Ullman [1, pp. 303-308].

Let f be the polynomial of degree n in $\mathbf{Z}_p[X]$ to be factored. The distinct degree factorization of f can be performed using $O((\log p)n^{2+\epsilon})$ operations in $\mathbf{Z}_p$ (see, for example, Ben-Or [4] for more details).

Consider factoring $g = f^{(d)}$ for a fixed $1 \leq d \leq n$. Our algorithm can construct the separating set $S = \{h_0, \ldots, h_{d-1}\}$ using $O((\log p)d)$ multiplications in R to compute the powers of x, and $O(d^{1+\epsilon})$ additions, multiplications and subtractions in R to compute the coefficients of $h(Y)$. This gives a total of $O((\log p)(md)^{1+\epsilon})$ operations in $\mathbf{Z}_p$ to compute S.

Now, for any partial factorization U and any polynomial v of degree $< m$, we can compute $Refine(U, v)$ with $O(m^{1+\epsilon})$ operations in $\mathbf{Z}_p$ by first computing v mod u for each $u \in U$, and then computing $\gcd(u, v \bmod u)$ for each $u \in U$. Therefore, each execution of the refinement step can be performed using $O((\log p)(md)^{1+\epsilon})$ operations in $\mathbf{Z}_p$. So to determine the complexity of our algorithm, we must get a bound on the number of times the refinement step is executed. If $p = 2$, the refinement step will be executed only once. Therefore, we can assume that p is odd.

Suppose that for some $1 \leq i < j \leq k$ the refinement step has been executed for $z = 0, \ldots, M$ and that $g_i g_j | u$ for some $u \in U$. Since S is a separating set, there is an $s \in S$ such that $\theta_i(s) = a$ and $\theta_j(s) = b$, where a and b are distinct elements of $\mathbf{Z}_p$. Then it follows that $\chi((a+z)(b+z)) = 1$ for $z = 0, \ldots, M$, where χ is the quadratic character on $\mathbf{Z}_p$. This allows us to obtain the following nontrivial bound on M.

Lemma 3.2. *Let p be an odd prime, and let $a, b \in \mathbf{Z}_p$, such that $a \neq b$ and*

$$\chi(ab) = \chi((a+1)(b+1)) = \cdots = \chi((a+M)(b+M)) = 1,$$

where χ is the quadratic character on $\mathbf{Z}_p$. Then $M < p^{1/2} \log p$.

Proof. Let $t = \lceil \frac{1}{2} \log p \rceil$. Let N be the number of solutions $(x, y_0, \ldots, y_{t-1}) \in \mathbf{Z}_p^{t+1}$ to the system of equations

$$(x+a+i)(x+b+i) = y_i^2 \qquad (i = 0, \ldots, t-1).$$

We will first show that

$$N \leq p + p^{1/2}(2^t(t-1)+1). \tag{1}$$

Now, for fixed $c \in \mathbf{Z}_p$ the number of $y \in \mathbf{Z}_p$ satisfying the equation $y^2 = c$ is precisely $1 + \chi(c)$.

Therefore,

$$N = \sum_{x \in \mathbf{Z}_p} \prod_{i=0}^{t-1} (1 + \chi((x+a+i)(x+b+i)))$$
$$= \sum_{0 \le e_0, \ldots, e_{t-1} \le 1} \sum_{x \in \mathbf{Z}_p} \chi\left(\prod_{i=0}^{t-1} (x+a+i)^{e_i}(x+b+i)^{e_i}\right).$$

In this last expression, the term corresponding to $e_0 = \cdots = e_{t-1} = 0$ is p.

Now let $e_0, \ldots, e_{t-1}$ be fixed with $l > 0$ of the e_i's are nonzero, and let $\lambda(X) = \prod_{i=0}^{t-1}(X+a+i)^{e_i}(X+b+i)^{e_i}$. We claim that $\lambda(X)$ is not a perfect square in $\mathbf{Z}_p[X]$. Suppose that it were. Then for distinct $i_1, \ldots, i_l$ between 0 and $t-1$, we would have

$$a + i_1 = b + i_2, \quad a + i_2 = b + i_3, \quad \ldots, \quad a + i_{l-1} = b + i_l, \quad a + i_l = b + i_1.$$

Summing, we have $la + \sum_\nu i_\nu = lb + \sum_\nu i_\nu$. But this implies that $la = lb$, and since $0 < l < p$, we can cancel, obtaining $a = b$, a contradiction. Therefore, $\lambda(X)$ is not a perfect square.

From Weil's Theorem (see Schmidt [16, p. 43]), for any monic polynomial λ in $\mathbf{Z}_p[X]$ that is not a perfect square, we have

$$\left|\sum_{x \in \mathbf{Z}_p} \chi(\lambda(x))\right| \le (r-1)p^{1/2},$$

where r is the number of distinct roots of λ in its splitting field. It follows that

$$N \le p + p^{1/2} \sum_{l=1}^{t} \binom{t}{l}(2l-1)$$
$$= p + p^{1/2}(2^t(t-1)+1).$$

This proves (1).

Now, the number of $x \in \mathbf{Z}_p$ such that

$$\chi((x+a+i)(x+b+i)) = 1 \qquad (i = 0, \ldots, t-1)$$

is at most $N/2^t$. The worst possible case is when all such x are bunched together near zero. So we have $M < N/2^t + t$. By (1), we have $M < p/2^t + p^{1/2}(t - 1 + 2^{-t}) + t$. Since $t = \lceil \frac{1}{2} \log p \rceil$, we have $M < p^{1/2} + \frac{1}{2} p^{1/2} \log p + \frac{1}{2} \log p + 2$. The right hand side of this inequality is asymptotic to $\frac{1}{2} p^{1/2} \log p$, and is less than $p^{1/2} \log p$ for $p > 16$. For $p < 16$, $p^{1/2} \log p > p$, and so the lemma is trivially true in this case. ∎

We see then that g can be factored with $O(p^{1/2}(\log p)^2(md)^{1+\epsilon})$ operations in $\mathbf{Z}_p$. Since this holds for each $1 \le d \le n$, it follows that f can be factored using $O(p^{1/2}(\log p)^2 n^{2+\epsilon})$ operations in $\mathbf{Z}_p$, which proves Theorem 1.

Remark 1. The idea of factoring a polynomial by examining the elements of the form $(s+z)^{(p-1)/2}$ where s is in the Berlekamp subalgebra and $z = 0, 1, 2$, etc., originates with Berlekamp [5, p. 732]. However, prior to this research, apparently no analysis has been done on the worst-case or average-case complexity of algorithms based on this idea.

Remark 2. Actually, there is a slightly more complicated version of our algorithm that runs in time $O((\log p)n^{2+\epsilon} + p^{1/2}(\log p)^2 n^{3/2+\epsilon})$. We'll briefly sketch this algorithm here, but we won't discuss it in detail because its running time is still essentially quadratic in n, and its average case running time does not appear to be as good as that of the algorithm in Section 2. To factor $g = f^{(d)}$, this algorithm computes a separating set S just as in Section 2, initializes U to $\{g\}$, and then does the following for each $s \in S$:

> Initialize z to 0. While $s \bmod u \notin \mathbf{Z}_p$ for some $u \in U$, replace U with $Refine(U, s+z)$, and then, if p is odd, replace U with $Refine(U, (s+z)^{(p-1)/2} - 1)$, and then increment z.

It is straightforward to show that the time required by this method to completely factor g is $O((\log p)dm^{1+\epsilon} + p^{1/2}(\log p)^2 \min(d,k)m^{1+\epsilon})$.

Remark 3. In some applications, one only requires a single irreducible factor of f. A slight variation of the algorithm in Remark 2 extracts a single irreducible factor of f and runs in time $O((\log p)n^{2+\epsilon} + p^{1/2}(\log p)^2 n^{1+\epsilon})$. In particular, an irreducible factor of a polynomial g that is the product of k distinct monic irreducible polynomials each of degree d can be extracted deterministically in time $O((\log p)dm^{1+\epsilon} + p^{1/2}(\log p)^2 m^{1+\epsilon})$, where $m = kd$.

4. Average-Case Analysis

In this section, we study the average case complexity of the our algorithm, assuming that the polynomial to be factored is chosen from a uniform distribution on all monic polynomials in $\mathbf{Z}_p[X]$ of degree n. Recall that to factor f, our algorithm first obtains a distinct degree factorization $f^{(1)}, \ldots, f^{(n)}$. To factor $f^{(d)}$, it executes the refinement step some number of times, say K_d times. In section 3, we proved that $K_d \le p^{1/2}(\log p)$. We might expect that on average, K_d is much less than than this. In this section, we will study the probability B that f is "bad" in the sense that $K_d > t$ for some $1 \le d \le n$, where $t = \lceil \log p \rceil$. We shall prove

Theorem 2. *Let f be a polynomial chosen from a uniform distribution on all monic polynomials of degree n in $\mathbf{Z}_p[X]$. Let B be defined as in the previous paragraph. Then $B = O((n \log p)^2/p)$.*

This theorem shows that our algorithm runs in polynomial time on all but at most an exponentially small fraction of the polynomials of degree n over $\mathbf{Z}_p$. The following is an immediate consequence of Theorems 1 and 2.

Corollary. *Let f be a polynomial chosen from a uniform distribution on all monic polynomials of degree n in $\mathbf{Z}_p[X]$. Then the expected running time of our algorithm is polynomial in n and $\log p$.*

We now prove Theorem 2. We can partition the polynomials of degree n in $\mathbf{Z}_p[X]$ according to their "factorization pattern." The factorization pattern π of a polynomial f is an n-tuple $(k_1, \ldots, k_n)$ where k_d is the number of irreducible factors (counting multiplicities) of degree d that divide f. Let B_π be the conditional probability that f is "bad" given that its factorization pattern is π. We will show that $B_\pi = O((n \log p)^2/p)$, from which Theorem 2 follows immediately.

We can write $B_\pi \leq B_{\pi,1} + \cdots + B_{\pi,n}$ where $B_{\pi,d}$ is the probability that $K_d > t$ given that the factorization pattern is π. Let's fix $1 \leq d \leq n$ for the moment, and let $k = k_d$. We shall prove that

$$B_{\pi,d} = O((k \log p)^2/p). \tag{2}$$

To prove this, we will need the following lemma.

Lemma 4.1. *Let p be an odd prime, χ be the quadratic character on $\mathbf{Z}_p$, and $t = \lceil \log p \rceil$. Then the number of pairs $(a, b) \in \mathbf{Z}_p^2$ such that $\chi(a + i) = \chi(b + i)$ for $i = 0, \ldots, t - 1$ is no more than $p(\log p)^2$.*

Proof. The number J of such pairs is no more than t plus the number J' of pairs (a, b) such that $\chi((a + i)(b + i)) = 1$ for $i = 0, \ldots, t - 1$. Now, J' is the number of pairs (a, b) for which there exist nonzero $c_1, \ldots, c_t$ in $\mathbf{Z}_p$ such that

$$\begin{aligned} ab &= c_1^2 \\ (a+1)(b+1) &= c_2^2 \\ &\vdots \\ (a+t-1)(b+t-1) &= c_t^2. \end{aligned} \tag{3}$$

Let N be the number of solutions $(a, b, c_1, \ldots, c_t) \in \mathbf{Z}_p^{t+2}$ to (3). We want to get a good upper bound on N. We have

$$\begin{aligned} N &= \sum_{a,b \in \mathbf{Z}_p} (1 + \chi(ab)) \cdots (1 + \chi((a+t-1)(b+t-1))) \\ &= \sum_{0 \leq e_1, \ldots, e_t \leq 1} \sum_{a,b \in \mathbf{Z}_p} \chi(a^{e_1} b^{e_1} \cdots (a+t-1)^{e_t} (b+t-1)^{e_t}) \\ &= \sum_{0 \leq e_1, \ldots, e_t \leq 1} \left(\sum_{a \in \mathbf{Z}_p} \chi(a^{e_1} \cdots (a+t-1)^{e_t}) \right) \left(\sum_{b \in \mathbf{Z}_p} \chi(b^{e_1} \cdots (b+t-1)^{e_t}) \right). \end{aligned}$$

In this last expression, the term corresponding to $e_1 = \cdots = e_t = 0$ is p^2. We can again use Weil's Theorem to bound the magnitude of each of the other terms, obtaining

$$\begin{aligned} N &\leq p^2 + p \sum_{l=1}^{t} \binom{t}{l} (l-1)^2 \\ &= p^2 + p \left(t(t-1)2^{t-2} - t2^{t-1} + 2^t - 1 \right). \end{aligned}$$

We divide this quantity by 2^t to obtain a bound on the number of (a, b) for which there exist nonzero $c_1, \ldots, c_t$ satisfying (3). Using the fact that $J \leq t + J'$, we have

$$J \leq p\left(\frac{t}{p} + \frac{p}{2^t} + \frac{t(t-1)}{4} - \frac{t}{2} + 1 - \frac{1}{2^t}\right).$$

The right hand side of this inequality is asymptotic to $\frac{1}{4}p(\log p)^2$ as $p \to \infty$, and some calculations show that it is less than $p(\log p)^2$ for all $p \geq 3$. ■

Now to prove (2). It will be convenient to let $\tilde{f}^{(d)}$ denote the product of *all* monic irreducible factors of f of degree d (including multiplicities). We can regard f as being chosen from a uniform distribution on all monic polynomials with factorization pattern π, and $\tilde{f}^{(d)}$ as being chosen chosen from a uniform distribution on all monic polynomials with k irreducible factors of degree d. Note that k is an upper bound on the number of irreducible factors of $f^{(d)}$, since we're assuming that the distinct degree factorization procedure removes multiplicities.

Let's say that two elements $a, b \in \mathbf{Z}_p$ are *indistinguishable* if $\chi(a+i) = \chi(b+i)$ for $i = 0, \ldots, t-1$. Let's say that two polynomials in $\mathbf{Z}_p[X]$ of equal degree are *indistinguishable* if each pair of corresponding coefficients are indistinguishable.

Now, $B_{\pi,d}$ is no greater than the probability that $\tilde{f}^{(d)}$ is divisible by two indistinguishable monic irreducible polynomials. This latter probability is no greater than k^2 times the probability that a randomly chosen pair of irreducible polynomials of degree d are indistinguishable. Let B' be this latter probability.

Let $I(d)$ be the number of indistinguishable pairs of monic irreducible polynomials of degree d. From Lemma 4.1, we see that $I(d) \leq (p(\log p)^2)^d$. Let $N(d)$ be the number of monic irreducible polynomials of degree d. As is well known (see, e.g., Rabin [15, Lemma 2]), $N(d) = \Theta(p^d/d)$. Then $B' = I(d)/(N(d))^2 = O(((\log p)^2/p)^d d^2) = O((\log p)^2/p)$. (2) now follows immediately, and so Theorem 2 is proved.

References

1. A. Aho, J. Hopcroft and J. Ullman. *The Design and Analysis of Computer Algorithms*, Addison-Wesley (1974).

2. E. Bach. "Realistic analysis of some randomized algorithms," in *Proc. 19th Annual ACM Symp. on Theory of Computing*, pp. 453-461 (1987).

3. E. Bach and V. Shoup. "Factoring polynomials using fewer random bits," Computer Sciences Technical Report No. 757, University of Wisconsin–Madison; *Journal of Symbolic Computation*, to appear (1988).

4. M. Ben-Or. "Probabilistic algorithms in finite fields," in *Proc. 22nd Annual Symp. on Foundations of Computer Science*, pp. 394-398 (1981).

5. E. Berlekamp. "Factoring polynomials over large finite fields," *Mathematics of Computation*, Vol. 24, No. 111, pp. 713-735 (1970).

6. A. Borodin and I. Munro. *The Computational Complexity of Algebraic and Numeric Problems,* American Elsevier (1975).

7. P. Camion. "Improving an algorithm for factoring polynomials over a finite field and constructing large irreducible polynomials," *IEEE Transactions on Information Theory,* Vol. 29, No. 3, pp. 378-385 (1983).

8. D. Cantor and E. Kaltofen. "Fast multiplication of polynomials over arbitrary rings," Department of Computer Science Technical Report No. 87-35, Rensselaer Polytechnic Institute (1987).

9. D. Cantor and H. Zassenhaus. "A new algorithm for factoring polynomials over finite fields," *Mathematics of Computation,* Vol. 36, No. 154, pp. 587-592 (1981).

10. G. Collins. "Factoring univariate integral polynomials in polynomial average time." *Eurosam '79: Springer-Verlag Lecture Notes in Computer Science #72,* pp. 317-329 (1979).

11. D. Coppersmith and S. Winograd. "Matrix multiplication via Behrend's method," *Proc. 19th Ann. ACM Symp. on Theory of Computing,* pp. 1-6 (1987).

12. J. von zur Gathen. "Factoring polynomials and primitive elements for special primes," *Theoretical Computer Science,* Vol. 52, pp. 77-89 (1987).

13. J. Hafner and K. McCurley. "On the distribution of running times of certain integer factoring algorithms," preprint (1987).

14. D. Knuth and L. Trabb Pardo. "Analysis of a simple factorization algorithm," *Theoretical Computer Science,* Vol. 3, pp. 321-348 (1976).

15. M. Rabin. "Probabilistic algorithms in finite fields," *SIAM J. Comput.*, Vol. 9, No. 2, pp. 273-280 (1980).

16. W. Schmidt. *Equations over Finite Fields,* Springer-Verlag Lecture Notes in Mathematics No. 536 (1976).

17. A. Schönhage. "Schnelle Multiplikation von Polynomen über Körpern der Charakteristik 2," *Acta Informatica,* Vol. 7, pp. 395-398 (1977).